Manipulacija konekcijom i čuvanje konekcija

Programski jezik Python ima veoma moćne alate za manipulaciju bazama podataka. On podržava veliki broj SUBP kao što su MySQL, Oracle SQL, Sybase, PostgreSQL, NoSQL, SQLite i mnogi drugi. U nastavku ćemo prikazati kratak opis i istorijat ovih rešenja.

- **MySQL** je sistem za upravljanje relacionim bazama podataka otvorenog koda. Kreirali su ga, 1995. godine, David Axmark i Michael Windenius u okviru firme MySQL AB, a danas na njega prava polaže korporacija Oracle. Prefiks *my* u nazivu inspirisan je imenom ćerke suosnivača Michaela. Ovaj jezik koristi klijent–server arhitekturu i uglavnom se koristi za web aplikacije. Pogodan je za skladištenje log-in informacija, e-commerce podataka i sl.
- **Oracle SQL** je prvi SUBP koji je dizajniran u svrhe obavljanja umreženih aktivnosti u okviru kompanije. Predstavlja najefektivniji i najjeftiniji način za upravljanje informacijama i aplikacijama. Kreirali su ga, 1977. godine, Larry Ellison, Boba Miner i Eda Outs, u okviru firme Software Development Laboratories (SDL). Naziv Oracle potiče od naziva koda u CIA projektu na kojem je Ellison radio.
- **Sybase** je nekada bilo softversko preduzeće koje se bavilo upravljanjem i analizom podataka u relacionim bazama. Inicijalni razvoj Sybasea pokrenuli su Robert Epstein, Mark Hoffman, Jane Doughty i Tom Haggin, 1984. godine u Berkliju. Interesantno je da su pregovori ove firme sa Microsoftom o licenciranju servera podataka doveli do izgradnje Microsoft SQL Servera 1.0. Ovaj SUBP se najčešće koristi u industriji finansija. Od 2010. godine na njega prava polaže SAP.
- **PostgreSQL** je podrazumevani sistem za upravljanje bazama podataka na macOS serverima, ali se može koristiti i na drugim operativnim sistemima. Nastao je od reči *Ingres*, koja predstavlja naziv relacione baze podataka namenjene komercijalnim i državnim svrhama. Osnivač Michael Stonebraker je 1985. godine odlučio da nastavi taj projekat, što je dovelo do nastanka PostgreSQL-a. Koristi se za mnoge web, mobilne, geoprostorne i analitičke aplikacije.
- **NoSQL** se, kao što sugeriše prefiks *No* u nazivu ovog SUBP-a, odnosi na nerelacione baze podataka. Koristi mehanizam za skladištenje i preuzimanje podataka koji se modelira drugačije od tabelarnih odnosa. Razvio ga je 1998. godine Carlo Strozzi. Najčešće se koristi za upravljanje velikim bazama podataka i za aplikacije koje skladište i koriste podatke u realnom vremenu.
- **SQLite** je sistem za upravljanje relacionim bazama podataka koji je ugrađen u program. Ne koristi klijent–server arhitekturu i ne zahteva prethodnu instalaciju za korišćenje. Kreirao ga je D. Richard Hipp 2000. godine u svrhe kontrole štete u američkoj mornarici. Pogodan je za male baze podataka i testiranje jednostavnih aplikacija.

Za svaku bazu podataka u Pythonu koju odaberemo koristimo odgovarajući interfejs. Većina SUBP koje Python podržava koristi standardni Python DB-API interfejs i funkcioniše u skladu sa njegovim definisanim standardima.

Radi jednostavnijeg objašnjenja modula, u početku ćemo koristiti sqlite3 modul koji je ugrađen u Python programski jezik, a zatim ćemo u nastavku pokazati kako iste funkcionalnosti izgledaju u mysql modulu. Modul sqlite3 ne zahteva prethodne instalacije niti dodatno konfigurisanje i konektovanje na server pre korišćenja, već pruža direktan pristup bazi. Još jedna od njegovih prednosti je otvoren kod koji je interportabilan na različitim

Copyright © Link group

platformama i operativnim sistemima. Reč *Lite* u nazivu ovog SUBP i sama govori da je reč o maloj bazi podataka koja ne troši puno resursa i nema velike zahteve po pitanju održavanja.

Dakle, sve što treba da uradimo da bismo imali pristup ovom ugrađenom modulu je da ga uvedemo u svoj program. Kao i do sada, komanda `import sqlite3` omogućava nam pristup funkcionalnostima tog modula. Za dalji rad sa podacima neophodna nam je baza podataka.

Konekcija sa bazom podataka u sqlite3 modulu

Pristup bazi podataka omogućava se preko konekcije. Osnovna metoda koju koristimo za uspostavljanje konekcije je `connect()`. Ova metoda na osnovu prosleđenog ulaznog parametra, koji predstavlja našu bazu, vraća objekat konekcije, odnosno klase `Connection`. Kreirani objekat konekcije neophodno je sačuvati u nekoj promenljivoj, koja će se u našem primeru zvati *conn*. Metodi konekcije prosledićemo bazu podataka *example.db*, koja još uvek nije kreirana. Kada metodi konekcije prosledimo nepostojeću bazu podataka, kreira se nova baza podataka pod tim imenom, što će se u našem slučaju i dogoditi.

Primer konekcije baze podataka korišćenjem sqlite3 modula:

```
import sqlite3

conn = sqlite3.connect('example.db')
```

Kreirani objekat `conn` sada predstavlja našu konekciju sa bazom i njime dalje možemo manipulisati. Takođe, kada je baza koja je prosleđena metodi `conection()` već postojeća u memoriji, neće se kreirati nova, već će se operacije vršiti nad postojećom bazom.

Modul sqlite3 nudi i mogućnost skladištenja baze podataka na glavnoj (RAM) memoriji. Kao što smo imali prilike da vidimo u lekciji *Čuvanje i transport podataka*, ova memorija omogućava najbrži pristup podacima, što omogućava neometano testiranje funkcionalnosti. Međutim, treba imati u vidu da ova memorija nije postojana, što znači da se sa završetkom našeg programa gube i svi podaci iz naše baze. Komanda kojom se omogućava konekcija sa bazom privremeno smeštenom u RAM memoriji je `conn = sqlite3.connect(:memory:)`.

Objekti konekcije podržavaju sledeće metode za manipulaciju:

- **`close()`** – Ova metoda onemogućava dalju upotrebu konekcije. U slučaju da neka operacija na bazi bude pokušana, pojaviće se greška. Isto važi i za sve kursor objekte koji pokušavaju da koriste konekciju. Zatvaranje konekcije pre izvršenja svih operacija za koje je zadužena funkcija `commit()` izazvaće vraćanje baze u stanje u kom je bila pre pokretanja transakcije pozivanjem `rollback()` metode.
- **`commit()`** izvršava sve zadate operacije u okviru transakcije. Ukoliko baza podataka podržava `auto-commit` funkciju, poželjno je njeno inicijalno postavljanje na *isključeno* i uvođenje metode koja će imati ulogu da time manipuliše. Moduli baza podataka koji ne podržavaju transakcije trebalo bi ovu metodu da uvedu bez povratne vrednosti.
- **`rollback()`** omogućava da se baza podataka vrati na početak bilo koje čekajuće transakcije. Zatvaranje konekcije bez prethodnog izvršenja svih operacija izazvaće pozivanje ove metode. Interesantno je da ova metoda zapravo nije obavezna, jer ne podržavaju svi moduli transakcije.

Copyright © Link group

- **`cursor()`** omogućava generisanje jednog ili skupa redova podataka i izvršavanje kompleksne logike nad njima. Ukoliko koristimo modul koji ne podržava kursor, modul će morati da oponaša pokazivače koristeći druga sredstva u skladu sa specifikacijom po kojoj je napisan.

Objekat kursor

Kursor predstavlja objekat `sql3.Cursor` klase. Ova klasa omogućava našem programu da upravlja tabelama i podacima iz njih. Objekat ove klase kreira se pozivanjem `cursor()` metode konekcijskog objekta, koji je takođe neophodno smestiti u promenljivu. U našem primeru, pozivanje kursora iz kreiranog objekta konekcije izgledaće ovako:

```
cursor = conn.cursor()
```

Objekat kursor podržava više metoda manipulacije, među kojima ćemo izdvojiti nekolicinu najčešće upotrebljavanih:

- **`close()`** je zadužena za manualno prekidanje objekta kursor. Sve operacije koje su pokušane nakon pozivanja ove funkcije dobiće poruku o grešci.
- **`execute(a[,b])`** prima operaciju u vidu parametra *a* i podatke u vidi parametra *b*. Parametar *b* je opcioni i može da prosledi sekvencu podataka ili da mapira podatke. Ovo je najefikasnija metoda za operacije koje se ponavljaju, jer kursor zadržava referencu na operaciju u slučaju da se ponovo prosledi isti objekat operacije.
- **`executemany(a, b)`** prima operaciju u vidu parametra *a* i obaveznu sekvencu podataka u vidu parametra *b*. Ova metoda upit iz parametra *a* izvršava nad skupom n-torki iz parametra *b* i daje jedan ili više rezultata. Povratna vrednost metode nije definisana.
- **`fetchone()`** vraća sledeći red skupa rezultata upita ili *null* ukoliko sledeći red ne postoji. Metoda javlja grešku kada metoda `execute()` koja joj prethodi nije definisana ili ne daje rezultate na osnovu prosleđenog upita.
- **`fetchmany(x)`** vraća sledeći set redova rezultata upita i prikazuje listu n-torki. Broj redova za preuzimanje je specificiran parametrom *x*. U suprotnom, matrica pokazivača sama određuje broj redova koji treba vratiti. Ukoliko je parametar *x* veći od broja redova u tabeli, vratiće se oni kojih ima. Metoda javlja grešku u istim slučajevima kao i `fetchone()`.
- **`fetchall()`** vraća sve redove koji odgovaraju zadatom uslovu iz upita. Metoda javlja grešku u istim slučajevima kao i prethodne dve metode.

Kreiranje baze podataka u sqlite3 modulu

Za kreiranje tabele podataka koristimo metodu `execute()`, koju pozivamo pomoću ranije definisanog objekta kursora *cursor*. Kao parametar ove metode prosledićemo SQL upit koji će se postarati da u bazi bude kreirana tabela sa atributima i ograničenjima koje definišemo. U nastavku ćemo prikazati kako kreiranje tabele izgleda na primeru lager liste proizvoda u prodavnici igračaka.

Primer kreiranja tabele korišćenjem `execute()` metode sqlite3 modula:

```
cursor.execute("""CREATE TABLE IF NOT EXISTS Toy_Store_Products(
  toy_id INT PRIMARY KEY,
```

Copyright © Link group

```
        product_name TEXT,
        price FLOAT,
        quantity INT);
        """)

    conn.commit()
```

Radi lakšeg razumevanja, prosleđeni upit ćemo podeliti na nekoliko celina:

- `CREATE TABLE` – obavezna naredba koja označava kreiranje tabele;
- `IF NOT EXISTS` – uslov koji zabranjuje ponovno kreiranje tabele ukoliko ona već postoji; ovaj uslov jeste opcion, međutim, njegovo izostavljanje će izazvati grešku: *'sqlite3.OperationalError: table Toy_Store_Products already exists'*;
- `Toy_Store_Products` – naziv tabele koju želimo da kreiramo.

U nastavku upita prosleđeni su atributi, odnosno kolone koje želimo da tabela ima. Svaki od atributa je definisan u formatu **`NAME DATA_TYPE CONSTRAINT`**, prema konvenciji pisanja upita. Sve tipove podataka koje SQLite podržava moguće je videti na ovom linku. Što se tiče ograničenja u našoj tabeli, na atribut šifre proizvoda smo postavili ograničenje `PRIMARY_KEY`, koje se stara za to da sve vrednosti tog atributa budu jedinstvene.

Nakon definisanja tabele podataka izvršićemo transakciju nad bazom pokretanjem naredbe `conn.commit()`. Kao što smo i ranije pominjali, ova naredba pravi trajne izmene u bazi podataka i ne dozvoljava da se operacije nad bazom izvrše delimično.

Izmeniti kod tako da se u bazi example.db napravite tabelu Book pomoću sqlite3 modula koja bi trebalo da sadrži atribute: book_id(ceo broj, predstavlja primarni ključ za ovu tabelu), writer(string), genre (string), year (ceo broj).

Radno okruženje

Pitanje

Komanda `sqlite3.connect('test.db')` vraća objekat:

- baze podataka
- **konekcije**
- n-torke podataka

Objašnjenje:

Metoda `sqlite3.connect('example.db')` vraća objekat klase `Connect` iliti konekcije.

Copyright © Link group

Konekcija sa bazom podataka u mysql modulu

U prethodnom primeru smo videli kako se baza podataka u sqlite3 modulu može konektovati i kreirati pomoću kursora. Konekcija sa MySQL bazama podataka ima nešto drugačiju rutinu. Naime, da bismo uspostavili konekciju sa bazom u ovom modulu, neophodno je da se povežemo sa MySQL serverom.

Pre nego što započnemo rad u ovom modulu, neophodno je da izvršimo njegovu instalaciju. To činimo pomoću naredbe `pip3 install mysql-connector-python`. Zatim ćemo instalirani modul uvesti u program komandom `import mysql.connector`, kako bismo mogli da ga koristimo. Sa ovako pripremljenim okruženjem možemo da započnemo konekciju na server.

Objekat konekcije dobijamo pozivanjem metode `connect()` iz našeg *mysql.connector* modula. S obzirom na to da se konektujemo na server, potrebno je da funkciji prosledimo parametre koji će konekciju usmeriti tamo gde je potrebno. Dva obavezna parametra konekcije su `username` i `database`. U ove parametre prosleđujemo korisničko ime sa kojim želimo da pristupimo bazi i naziv baze kojoj pristupamo. U našem slučaju, kao korisnika ćemo proslediti *root*, koji predstavlja *superkorisnika*, odnosno korisnika koji ima sve privilegije u bazi. Za ime baze ćemo uzeti našu *example.db* bazu podataka, s obzirom na to da ćemo rad u ovom modulu prikazati na istom primeru koji smo koristili i u sqlite3 modulu.

Funkcija `connect()` podržava veliki broj parametara koji nisu obavezni. Sve parametre koje je funkciji moguće proslediti možemo pronaći na ovom linku. Na listi parametara primećujemo da je za konekciju neophodno i unošenje lozinke, kao i servera i porta za konekciju. Međutim, primetićemo da ovi parametri poseduju podrazumevajuće vrednosti koje će biti automatski dodeljene ukoliko ih ne definišemo, što se dešava u našem slučaju.

Drugim rečima, funkcija `connect()` će za *password* postaviti prazan string; za server na koji se povezujemo, odnosno *host*, dodeliće podrazumevajuću adresu 127.0.0.1, koja predstavlja naš lokalni server, a kao vrednost porta automatski će biti postavljena vrednost 3306, koja je zapravo podrazumevajući port za MySQL. Konekciju ćemo potom sačuvati u promenljivu `conn` kao što je prikazano u primeru ispod.

Radno okruženje

```
import mysql.connector

mydb = mysql.connector.connect(
host="localhost",
user="root",
password="password123",
database="mydb"
)


cursor = mydb.cursor()

cursor.execute("""CREATE TABLE IF NOT EXISTS Toy_Store_Products(
  toy_id INT PRIMARY KEY,
  product_name TEXT,
  price FLOAT,
  quantity INT);
""")

mydb.commit()
```

Copyright © Link group

Objašnjenje:

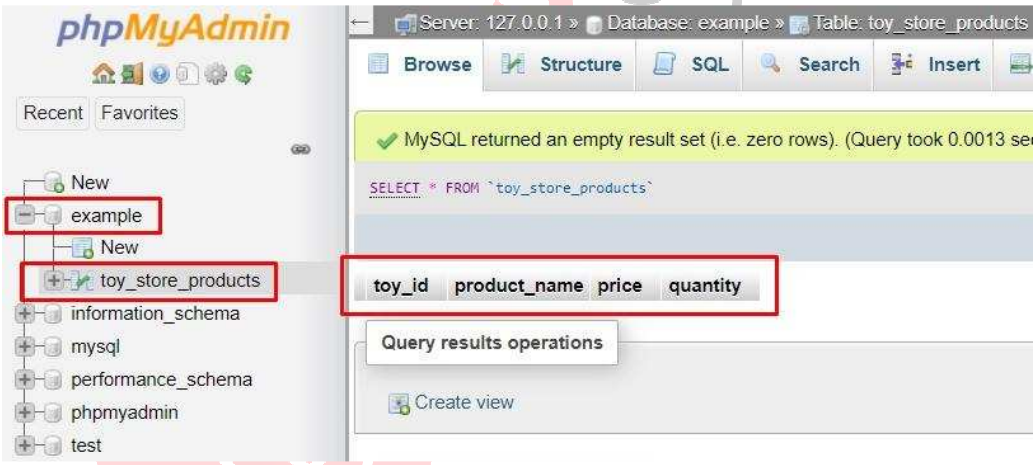
Rezultat ovako izvršenih komandi biće uspostavljena konekcija sa bazom podataka *mydb* na našem lokalnom serveru *localhost* gde je username root a šifra password123 . Baza podataka kojoj smo pristupili je trenutno prazna.

Za upis podataka u bazu potrebna nam je tabela koju ćemo kreirati putem upita na isti način kao i u prethodnom primeru sa sqlite3 modulom. Ne zaboravimo da je prethodno potrebno kreirati kursor koji nam omogućava pozivanje metode `execute()`, koja nam je neophodna za izvršavanje upita. Konačno, naredbom `commit()` izvršićemo trajne promene na bazi.

U radnom okruženju izmeniti kod tako što ćete napravite tabelu Student pomoću mysql.connector modula koja bi trebalo da sadrži atribute: student_id (ceo broj, predstavlja primarni ključ za ovu tabelu), first_name (string), last_name (string), city (string), department(string).

Radno okruženje

Primetićemo da je postupak kreiranja tabela u relacionim bazama podataka isti bez obzira na modul koji koristimo. Upit koji je prosleđen metodi `execute()` daje isti rezultat kao i onaj koji je prosleđen bazi podataka u sqlite3 modulu. Kao rezultat izvršenja transakcije dobićemo kreiranu tabelu na našem lokalnom serveru, što se može videti na slici 5.1).



Slika 5.1. Prikaz baze podataka kreirane u mysql modulu u phpMyAdmin okruženju

Rezime

- Programski jezik Python podržava veliki broj SUBP, kao što su MySQL, Oracle SQL, Sybase, PostgreSQL, NoSQL, SQLite i mnogi drugi.
- Svi navedeni sistemi se koriste za upravljanje relacionim bazama podataka, osim NoSQL-a, koji predstavlja nerelacioni model.
- SQLite jedini ne koristi klijent–server arhitekturu. Ovaj modul je ugrađen u program i ne zahteva povezivanje na server.
- Za konekciju sa bazom podataka koristimo metodu `connect()` klase `Connect` koja vraća objekat konekcije. Ova klasa podržava sledeće metode za manipulaciju:

Copyright © Link group

- `close()` – zatvaranje konekcije;
- `commit()` – izvršavanje transakcije;
- `rollback()` – poništavanje transakcije;
- `cursor()` – pokazivač koji služi za generisanje redova podataka.

- Osnovne metode koje klasa `Cursor` podržava su:

  - `close()` – zatvaranje kursora;
  - `execute()` – izvršavanje upita;
  - `executemany()` – izvršavanje upita koje vraća više od jednog rezultata;
  - `fetchone()` – vraća sledeći red tabele;
  - `fetchmany()` – vraća nekoliko redova tabele;
  - `fetchall()` – vraća sve redove tabele.

- U sqlite3 modulu dovoljno je da metodi konekcije u vidu parametra prosledimo naziv baze podataka na koju želimo da se konektujemo. Ukoliko baza ne postoji, kreiraće se nova sa tim imenom.
- U mysql modulu, kao obavezne parametre konekcije moramo proslediti username i database. Ako se ostali parameti ne proslede, podrazumeva se da lozinka korisnika ne postoji i da će se konekcija izvršiti na localhost na portu 3306, što je port namenjen za MySQL.
- Kreiranje nove tabele se u svim modulima vrši pomoću funkcije `execute()` i ključnih reči upita `CREATE TABLE`. Prateća izjava `IF NOT EXISTS` nije obavezna, ali je poželjna. Pri definiciji tabele neophodno je navesti njene atribute u sintaksi `NAME DATA_TYPE CONSTRAINT`.

Copyright © Link group